**ЛАБОРАТОРНА РОБОТА №2**

**Тема:** Сервіси та Dependency Injection

**Мета роботи:** ознайомитися зі структурою ASP.NET Code додатків, навчитися створювати найпростіші та використовувати наявні компоненти middleware у ASP.NET Core.

**Хід роботи:**

**Завдання 1.** Зробити обробку звернення до адреси “/services/list”, реалізувавши виведення інформації про всі сервіси, додані у додаток (тип сервісу, тип реалізації сервісу, життєвий цикл сервісу).

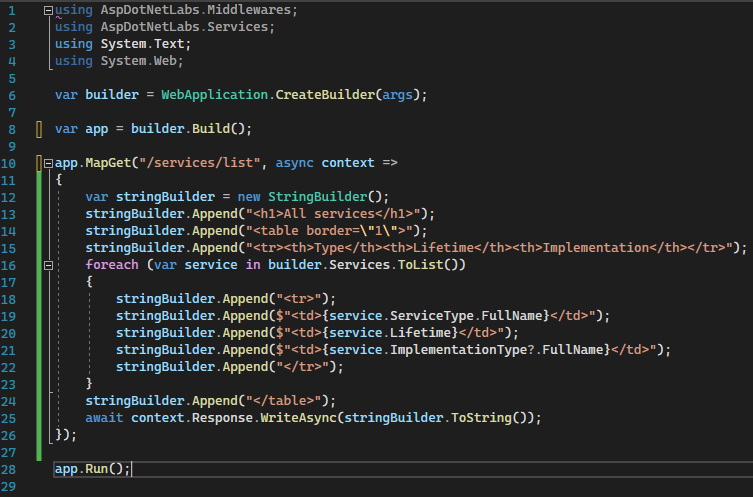


Рис. 1. Реалізація виведення таблиці всіх сервісів

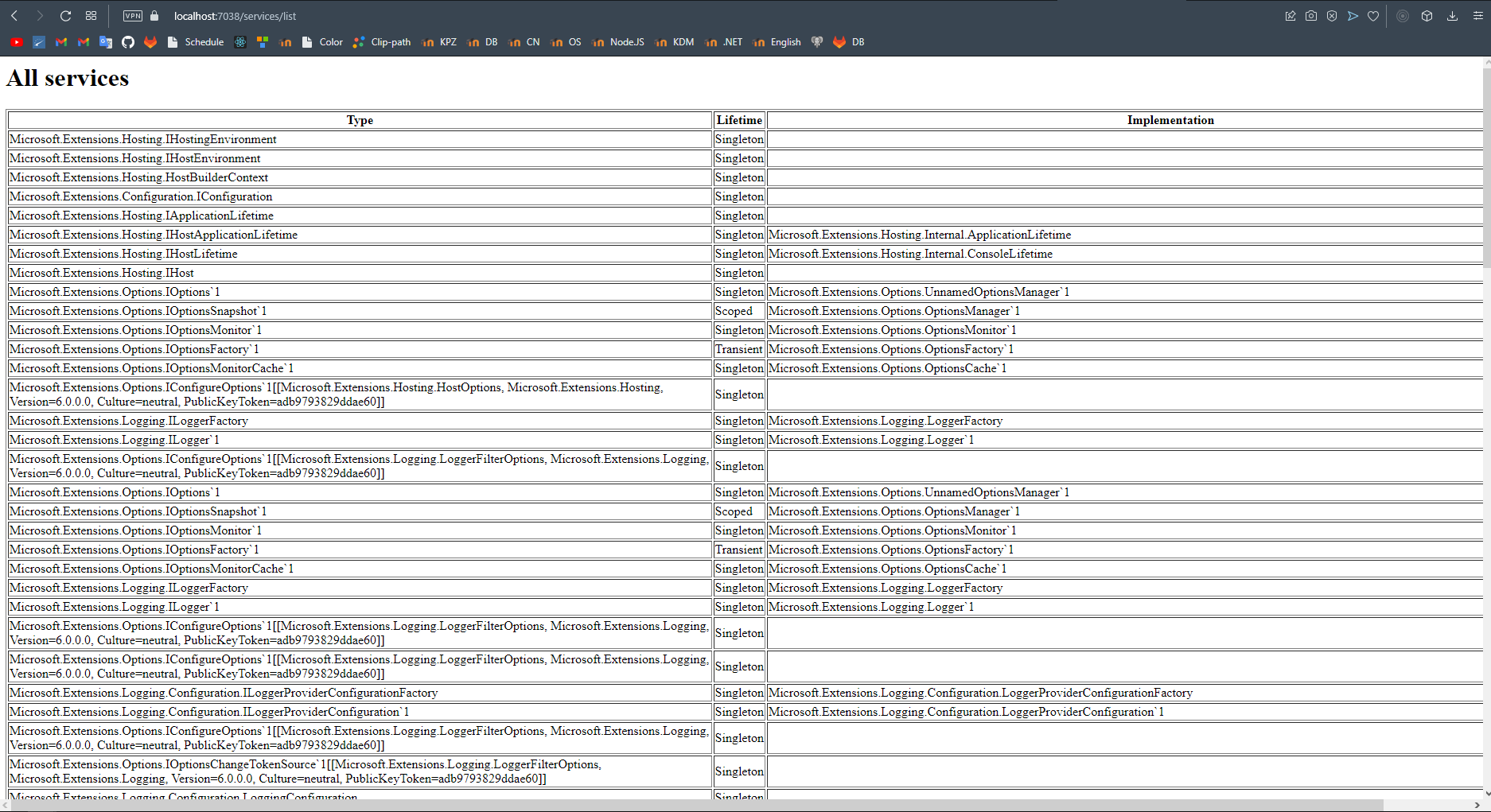


Рис. 2. Результат роботи за адресою /services/list

У звіті до лабораторної роботи навести коротку інформацію про кожний сервіс (достатньо одного речення, що чітко описує сервіс).

*Сервіс із життєвим циклом Transient буде створюватися кожен раз коли його інжектять в інші сервіси.*

*Сервіс із життєвим циклом Scoped буде створюватися один раз для запиту. Не залежно скільки разів його інжектили.*

*Сервіс із життєвим циклом Singleton буде створений при запуску серверу і надалі буде інжектитися у всі сервіси один і тей самий обєкт.*

**Завдання 2.** Створити три сервіси з різними життєвими циклами і продемонструвати їх роботу:

2.1) Тип “Transient”, назва “TimerService” - повертає поточну дату та час. Результати вивести за адресою “/services/timer”

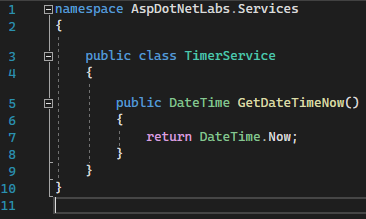


Рис. 3. Код класу TimerService

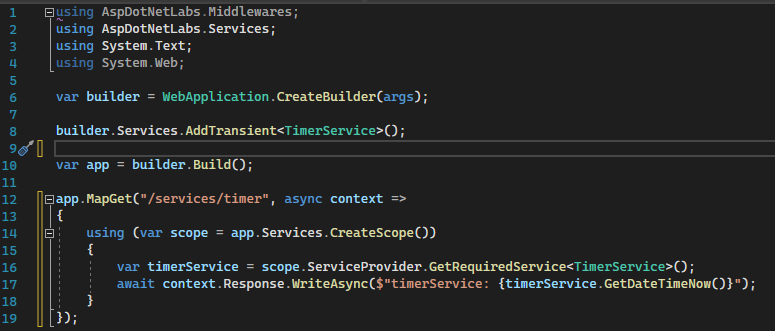


Рис. 4. Реєстрація TimerService в DI і використання у middleware

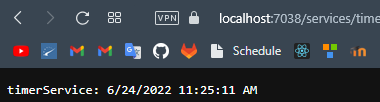


Рис. 5. Результат роботи за адресою /services/timer

2.2) Тип “Scoped”, назва “RandomService” - в конструкторі генерує випадкове число. За допомогою методу чи властивості повертає його. Результати роботи вивести за адресою “/services/random”. Має виводитися два числа, які співпадають (слайди 35-39 та 42-43 до лекції).

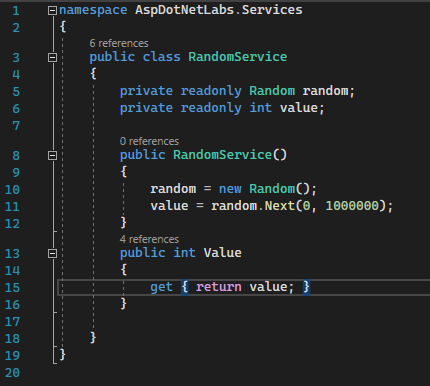


Рис. 6. Код класу RandomService



Рис. 7. Реєстрація RandomService в DI і використання у middleware

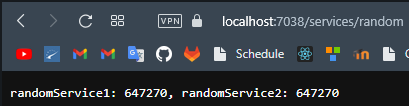


Рис. 8. Результат роботи за адресою /services/random

2.3) Тип “Singleton”, назва “GeneralCounterService” - веде підрахунок кількості звернень до кожної URL-адреси. Оскільки сервіс матиме тип “Singleton”, то підрахунок кількості буде вестися для усіх запитів. Результати вивести за адресою “/services/general-counter”

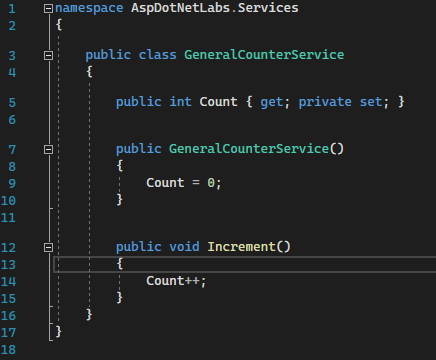


Рис. 9. Код класу GeneralCounterService



Рис. 10. Реєстрація GeneralCounterService в DI і використання у middleware

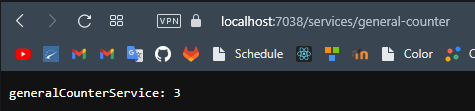


Рис. 11. Результат роботи за адресою /services/general-counter

**Завдання 3.** Для кожного сервісу створити відповідний middleware.

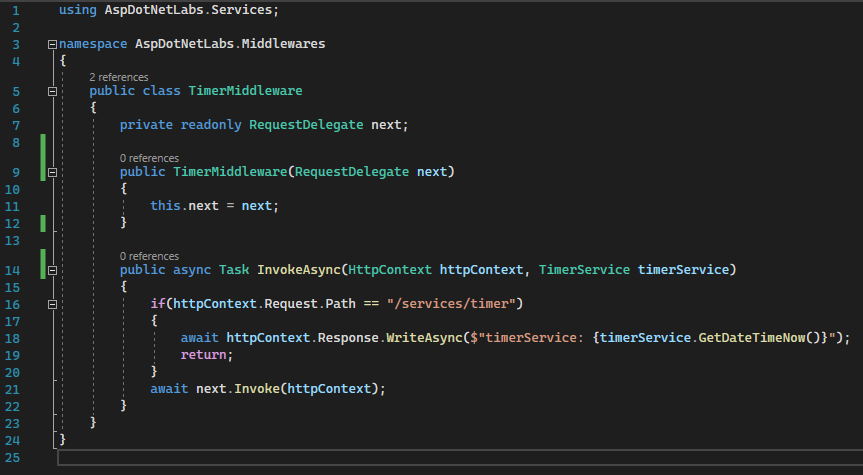


Рис. 12. Код класу TimerMiddleware

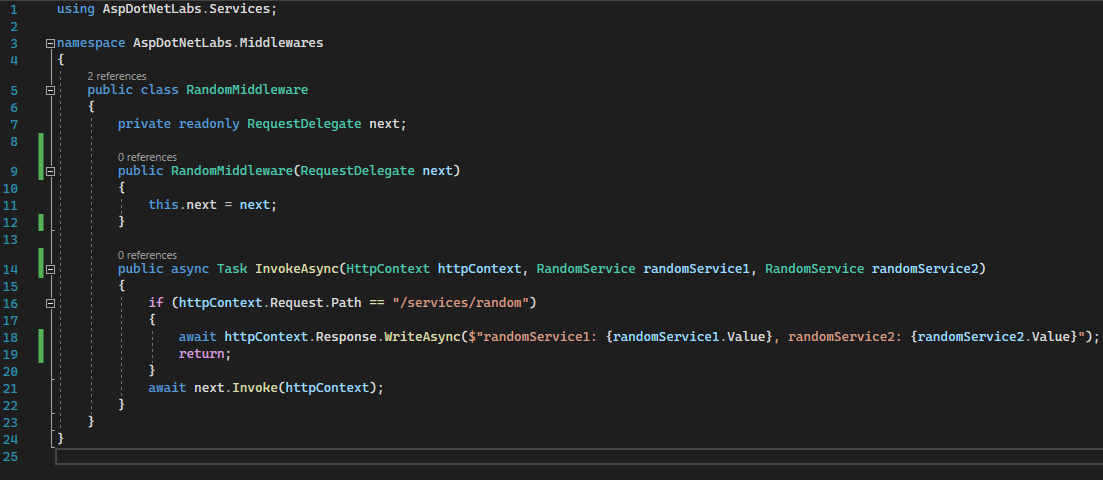


Рис. 13. Код класу RandomMiddleware

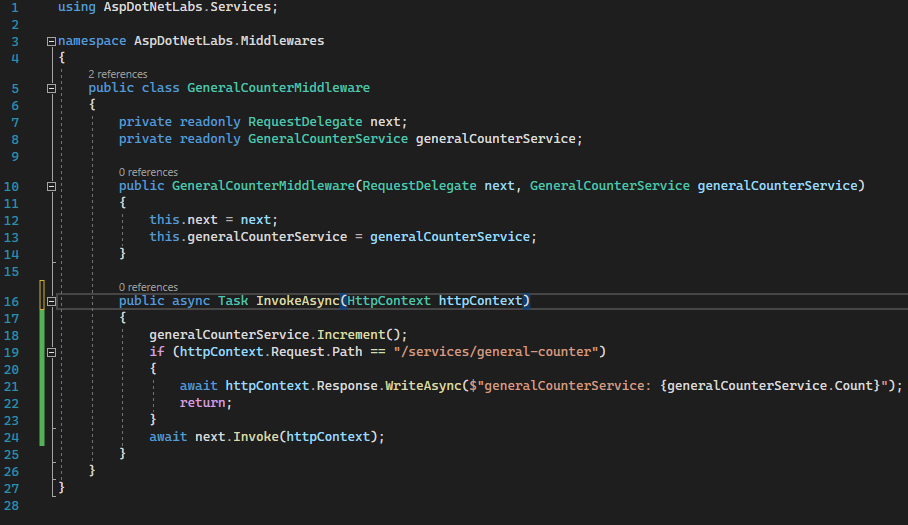


Рис. 14. Код класу GeneralCounterMiddleware

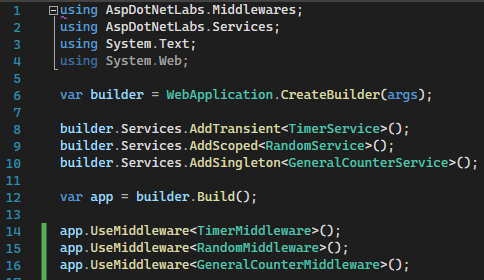


Рис. 15. Реєстрація middleware`ів

**Висновок:** на цій лабораторній роботи ми ознайомилися зі структурою ASP.NET Code додатків, навчитися створювати найпростіші та використовувати наявні компоненти middleware у ASP.NET Core.